Deen Dayal Upadhyaya College University of Delhi



Course: Bsc (Hons) Computer Science

Artificial Intelligence Practical File(LOCF)

Unique Paper Code: 32341601

**Submitted To: Submitted By:**

Dr. Anuja Soni Anupam Jena

Class Roll: 21HCS4119

Univ Roll : 21015570018

1. Write a prolog program to calculate the sum of two numbers.

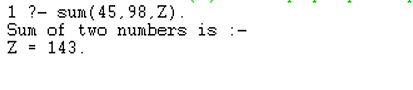
Code:

sum(X,Y,Z):-

write(‘Sum of two numbers is :-‘) ,

Z is X+Y .

Output:



2. Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.

Code:

max(A,B,M):-

A>B,

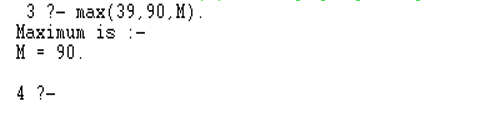
M is A;

A<B,

M is B,

write(‘Maximum is :-‘)

Output:



3. Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.

Code:

factorial(0,1).

factorial(N,F):-

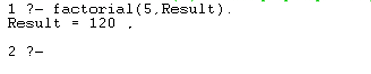
N>0,

N1 is N-1,

factorial(N1, F1),

F is N\*F1.

Output:



4. Write a program in PROLOG to implement generate\_fib(N,T) where T represents the Nth term of the fibonacci series.

Code:

fib(0,0) .

fib(1,1) .

fib(N,T) :-

N>1,

N1 is N-1,

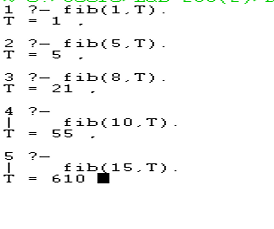
N2 is N-2,

fib(N1, T1),

fib(N2 , T2),

T is T1+T2.

Output:



5. Write a Prolog program to implement GCD of two numbers.

Code:

gcd(A,B,M):-

A=:=B,

M is A;

M is B.

gcd(A,B,M):-

A=0,

M is B.

gcd(A,B,M):-

B=0,

M is A.

gcd(A,B,M):-

A>B,

gcd(B,A,M).

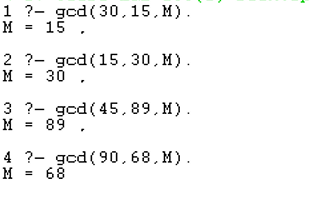
gcd(A,B,M):-

A<B,

T is B mod A,

gcd(A,T,M).

Output:



6. Write a Prolog program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.

Code:

power(\_ , 0 , 1).

power(Num , Pow ,Ans):-

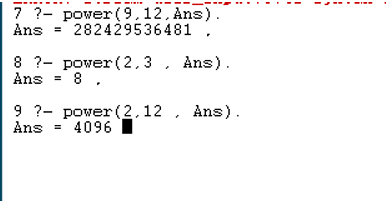
Pow>0,

Pow1 is Pow-1,

power(Num , Pow1 , Temp),

Ans is Temp\*Num.

Output:



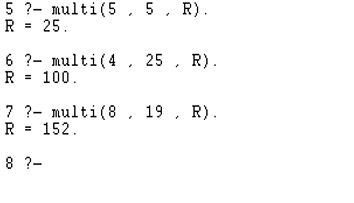
7. Prolog program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.

Code:

multi(N1 , N2 , R):-

R is N1\*N2.

Output:



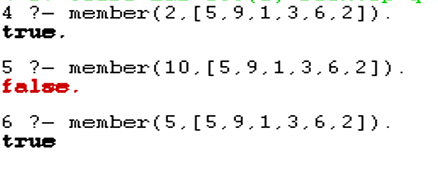
8. Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.

Code:

member(X , [X|T]).

member(X, [H|T]):- member(X, T).

Output:



9. Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.

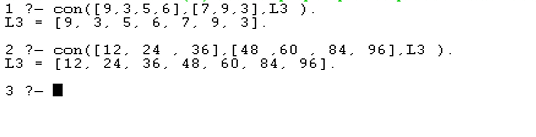
Code:

con([ ] , L , L).

con([H|L1] , L2 , [H|L3]):-

con(L1, L2 , L3).

Output:



10. Write a Prolog program to implement reverse (L, R) where List L is original and List R is reversed list.

Code:

con([ ] , L , L).

con([H|L1] , L2 , [H|L3]):-

con(L1, L2 , L3).

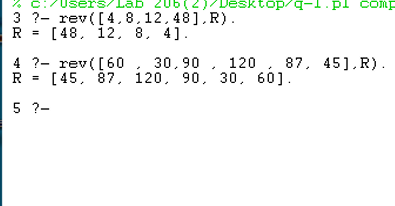
rev([ ] , [ ]).

rev( [H|T] , R):-

rev(T , Temp) ,

con(Temp , [H] , R).

Output:



11. Write a program in PROLOG to implement palindrome (L) which checks whether a list L is a palindrome or not.

Code:

con([ ] , L , L).

con([H|L1] , L2 , [H|L3]):-

con(L1, L2 , L3).

rev([ ] , [ ]).

rev( [H|T] , R):-

rev(T , Temp) ,

con(Temp , [H] , R).

compare([ ] , [ ]):-

write(‘It s a palindrome’).

compare( [X|L] , [X|R]):- compare(L, R).

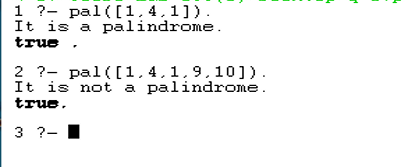
compare([X|L] , [Y|R]):- write(‘It is not a palindrome.’).

pal(L):-

rev(L , R),

compare(L , R).

Output:



12. Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.

Code:

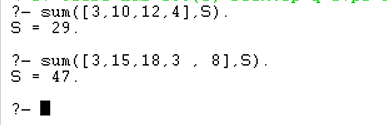
sum( [ ] , 0 ).

sum([H|T] , S):-

sum(T, Temp),

S is H+Temp.

Output:



13. Write a Prolog program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.

Code:

len([],0).

len([H|T],Count):-len(T,Temp),

Count is 1+Temp.

even\_odd(L):-len(L,Count),

A is mod(Count,2),

A=\= 0 ->

(

write('Length is odd')

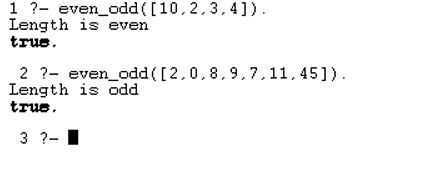
);

(

write('Length is even')

).

Output:



14. Write a Prolog program to implement nth\_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.

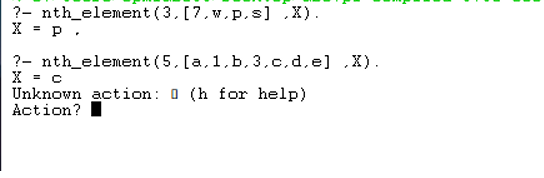
Code:

nth\_element(1, [H|T] , H).

nth\_element(N, [H|T] , X):- N1 is N-1,

nth element(N1 , T , X).

Output:



15. Write a Prolog program to implement maxlist(L, M) so that M is the maximum number in the list.

Code:

conc([] , L, L).

conc([H|L1] , L2 , [H|L3]):-

conc(L1 , L2 , L3).

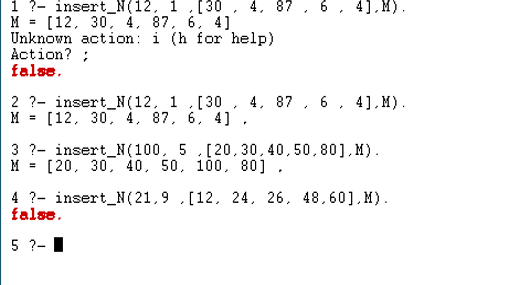
insert\_N(I , 1 , [X|Y] , M):- conc([I] , [X|Y], , M).

insert\_N(I , N , [X|Y]) : - N>1 ,

N1 is N-1,

insert\_N(I , N1 , Y, M).

Output:



16. Write a prolog program to implement insert\_nth (I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.

Code:

max(X,Y,Z):- X>Y,Z is X.

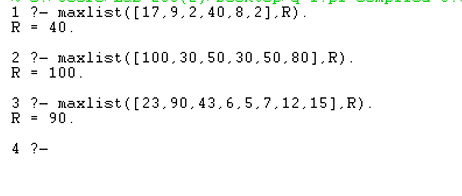
max(X,Y,Z):- X =< Y , Z s Y.

maxlist([] , 0) :- !.

maxlist([R] , R) :- !.

maxlist([H|T] , R) :- maxlist(T ,R1) , max(H , R1 , R) , !.

Output:



17. Write a Prolog program to implement delete\_nth (N, L, R) that removes the element on Nth position from a list L to generate a list R.

Code:

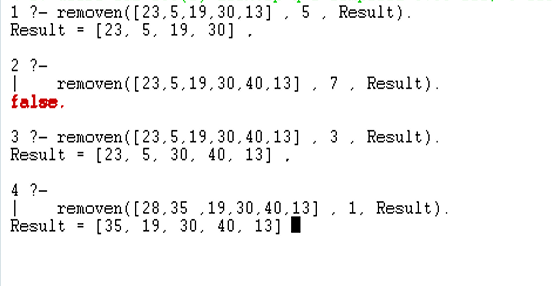
removen([\_ |List] , 1 , List),

removen([H|List] , Pos , [H | Result]):-

Pos1 is Pos-1,

removen(List , Pos1 , Result).

Output:



18. Write a program in PROLOG to implement merge (L1, L2, L3) where L1 is first ordered list and L2 is second ordered list and L3 represents the merged list.

Code:

merge(X , [] , X).

merge([] , Y, Y).

merge([X|X1] , [Y|Y1] , [X|Z] ):- X<Y,!,merge(X1,[Y|Y1],Z).

merge([X|X1] , [Y|Y1] , [X, Y|Z] ):- X=Y , !, merge(X1,Y1,Z).

merge([X|X1] , [Y|Y1] , [Y|Z]):- X>Y , ! , merge([X|X1] , Y1 , Z).

Output:

